EXAMPLE.TREE.1	
Create an attribute inside item itm of tree.	
the program checks if the item already has attributes: if it has, it inserts the new one at the bottom of the list ofattributes, otherwise it enters it as the first attribute of the item	
ATTRIBUTES	
`tree`	Tree pointer
`itm`	Item pointer
`sA`	Attrib search attribute
`t1`	Temporary attribute
`attP`	New attribute
CODE	

```
attP=TREE_ITM(tree,£NEW,£XIT,£ATTR);
TREE_ITM(tree,attP,£ITM,"varDbg");
TREE_ITM(tree,attP,£LBL,"varDbg");
TREE_ITM(tree,attP,£VAL,"100");

!!! Attributes list of item ;
sA=TREE_ITM(tree,itm,£ATT);

!!! If null, create new first item ;
#IF(sA<=0);
     TREE_ITM(tree,itm,£ATT,attP);
#ELSE;
     !!! Search last attributes ;
     #WHILE(1);
          t1=TREE_ITM(tree,sA,£NXT);
          #IF(t1<=0);
               #BREAK;
          #END;

          sA=t1;
     #END;

     TREE_ITM(tree,sA,£ADDA,attP);
#END;
```

EXAMPLE.DOC.1	
Generate ad hoc structured document	

This example shows as generate an ad hoc structure document.
It uses an odt (text) opendocument file as source.
This file includes 4 elements: a t_init table, to be used as a reference for inserting variable elements, and to be kept in the final file. 3 tables t_a, t_b and t_c to be cloned and inserted freely, to compose the desided file, and which we will eliminate in the final file.
The file we want to compose will have the structure: t_init, t_a_1, t_b_1_1, t_c_1_1_1, t_c_1_1_2, t_a_2, t_b_2_1, t_c_2_1_1, t_b_2_2, t_c_2_2_1 ...

ATTRIBUTES

`dstFn`	Destination filename
`dstDoc`	Pointer to destination document
`aDT`	Pointer to document table t_a
`bDT`	Pointer to document table t_b
`cDT`	Pointer to document table t_c
`phT`	Place holder doc table, where we are arrived with insert

CODE

```
!!! Generate a 'test.odt' file, starting from 'tpl.odt' : both in working
directory, where it is this PKA/PWK ;
dstFn=            DOC_TPL("tpl.odt",FS_DCWD()++"\"++"test.odt",FS_DCWD());

dstDoc=           DOC_DOC(£OO,dstFn);

!!! Load TBLs to use to shape doc ;
aDT=              DOC_TBL(dstDoc,"t_a");
bDT=              DOC_TBL(dstDoc,"t_b");
cDT=              DOC_TBL(dstDoc,"t_c");

phT=              DOC_TBL(dstDoc,"t_init");           !!! Initial position ;

phT=  DOC_TBLDUP(aDT,"t_a_1",2,£AFTER,phT);
phT=  DOC_TBLDUP(bDT,"t_b_1_1",2,£AFTER,phT);
phT=  DOC_TBLDUP(cDT,"t_c_1_1_1",2,£AFTER,phT);
phT=  DOC_TBLDUP(cDT,"t_c_1_1_2",2,£AFTER,phT);
phT=  DOC_TBLDUP(aDT,"t_a_2",2,£AFTER,phT);
phT=  DOC_TBLDUP(bDT,"t_b_2_1",2,£AFTER,phT);
phT=  DOC_TBLDUP(cDT,"t_c_2_1_1",2,£AFTER,phT);
phT=  DOC_TBLDUP(bDT,"t_b_2_2",2,£AFTER,phT);
phT=  DOC_TBLDUP(cDT,"t_c_2_2_1",2,£AFTER,phT);

!!! Delete original table templates from the doc ;
DOC_TBLRMV(aDT);
DOC_TBLRMV(bDT);
DOC_TBLRMV(cDT);

DOC_SAV(dstDoc);
DOC_CLS(dstDoc);
```

EXAMPLE.SOK.1	
Simple TCP listener	
How to make a simple code to acquire packages from multiple sources. The listening port is a constant. There is a VAR NET that contains 2 method : Listener, awaiting for connections, and trig, to manage the connections. Transmission use POWER-KI PKT protocol over TCP .	
ATTRIBUTES	
NET\lstnPort	Listener port
NET\exit	Exit condition for all methods
NET\Listener\sok	Pointer to socket
NET\Listener\trig	Pointer to trigger
NET\Trig\SOK	Socket pointer passed by listener
NET\Trig\ADDRESS	Address of the remote connection
NET\Trig\PORT	Port of the remote connection
NET\Trig\pck	Packet received
NET\Trig\res	Temporary result value

CODE

```
!!! Code of Listener ;
sok=  SOK_NEW( £TCP,0,port);

trg=  TRIG("\NET\Trig");
TRIGSET(trg,"SOK","SOK");
TRIGSET(trg,"ADDRESS","ADDRESS");
TRIGSET(trg,"PORT","PORT");

exit= 0;
SOK_LKW(sok,0,0,0,trg,£THREAD);
exit= 1;
TRASH(sok);
sok=  NULL;

!!! ---------------------------------------------- :
!!! Code of Trig ;
#WHILE(exit==0);
     res= SOK_INQ(SOK,1000);
     #IF(ISNULL(res));
          !!! Connection closed by remote peer, nothing more to do ;
          #BREAK;
     #END;

     #IF(res<=0);
          !!! Connection still active, but nothing transmitted ;
          #SKIP;
     #END;

     pck= SOK_RDS(sok,£PKT,NULL,1000); !!! Packet transmittedd with PKT prot.;
     #IF(~pck==0);
          !!! Anomaly in the comunication ;
          SOK_RDS(sok,£A,res,1000);    !!! Flush reception buffer ;
          #SKIP;
     #END;
```

```
     !!! Manage pkt HERE ;
#END;

!!! Not to trash SOK, the system will trash it ;
```

EXAMPLE.IEP.1	
IEP server	
This example describes how to realize a simple IEP server. The server manages 4 area, responding itf 100,200,300,400, prf 0. The areas have size 1000 bytes.	
ATTRIBUTES	
`iep`	IEP pointer
`buf1`	Area 1 buffer
`buf2`	Area 2 buffer
`buf3`	Area 3 buffer
`buf4`	Area 4 buffer
`runFlg`	Flag to set not to 1 to stop program
CODE	

```
!!! Initialize server pointer with IP address, responding port and protocol
type (IEP, industrial ethernet protocol) ;
iep=IEP_SRV("192.168.2.144",1800,£IEP);

!!! Define 4 responding areas, all sized 1000 bytes ;
!!! For each area, create a buffer of support for the data ;
!!! IEP areas adresses are ITF 100,200,300,400, PRF 0 ;

buf1=BUF_NEW(1000,£U8);
IEP_SRVADD(iep,100,0,buf1);

buf2=BUF_NEW(1000,£U8);
IEP_SRVADD(iep,200,0,buf2);

buf3=BUF_NEW(1000,£U8);
IEP_SRVADD(iep,300,0,buf3);

buf4=BUF_NEW(1000,£U8);
IEP_SRVADD(iep,400,0,buf4);

!!! Start IEP server ;
IEP_SRVCMD(iep,£START);

runFlg=1;

!!! Program loop until something external se runFlg not 1 ;
#WHILE(runFlg==1);
     SLEEP(100);
#END;

!!! Stop the server ;
IEP_SRVCMD(iep,£STOP);

!!! Destroy all pointers ;
TRASH(iep,buf1,buf2,buf3,buf4);
```

EXAMPLE.IEP.2	
IEP client connect to server	
This example describes hot to realize a software to connect to the previous IEP server.	

For each areas, it reads first I16 and if changed, write own timestamp at address 100.

ATTRIBUTES

iepT	IEP table
iep	Temporary IEP pointer
buf	Temporary buffer
c1	Counter
t1	Temporary attribute

CODE

```
!!! This table is oriented to store the 4 areas IEP pointer, buffer pointer and
last counter value ;
iepT=TBL_NEW(NULL,4,";",NULL,"IEP;BUF;oldCnt");

!!! For every area, set responding IP, port, protocol ;
!!! Set the comunication settings: ITF 100,200,300,400, PRF 0 ;
!!! Area sizws are 200 bytes ;

iep=IEP_CLI("192.168.2.144",1800,£IEP);
IEP_CLICON(iep,100,0);
IEP_CLIDAT(iep,0,0,1,200,0);
buf=BUF_NEW(1000,£U8);
IEP_CLIBUF(iep_buf);

!!! Save pointer inside iepT table ;
TBL_ITM(iepT,£IEP,1,iep);
TBL_ITM(iepT,£BUF,1,buf);

iep=IEP_CLI("192.168.2.144",1800,£IEP);
IEP_CLICON(iep,200,0);
IEP_CLIDAT(iep,0,0,1,200,0);
buf=BUF_NEW(1000,£U8);
IEP_CLIBUF(iep_buf);

TBL_ITM(iepT,£IEP,2,iep);
TBL_ITM(iepT,£BUF,2,buf);

iep=IEP_CLI("192.168.2.144",1800,£IEP);
IEP_CLICON(iep,300,0);
IEP_CLIDAT(iep,0,0,1,200,0);
buf=BUF_NEW(1000,£U8);
IEP_CLIBUF(iep_buf);

TBL_ITM(iepT,£IEP,3,iep);
TBL_ITM(iepT,£BUF,3,buf);

iep=IEP_CLI("192.168.2.144",1800,£IEP);
IEP_CLICON(iep,400,0);
IEP_CLIDAT(iep,0,0,1,200,0);
buf=BUF_NEW(1000,£U8);
IEP_CLIBUF(iep_buf);

TBL_ITM(iepT,£IEP,4,iep);
TBL_ITM(iepT,£BUF,4,buf);

runFlg=1;

!!! Run until something external set runFlg not 1 ;
#WHILE(runFlg==1);
```

```
     !!! Loop 4 areas, reading all, ... ;
     !!! checking if the counter at address 0 (0 bsd) is changed ... ;
     !!! since last run. If changed, write clock at address 100 (0 bsd) ;
     c1=0;
     #WHILE(c1+=1 <=4);
          iep=TBL_ITM(iepT,£IEP,c1);
          t1=IEP_CLIRD(iep);
          #IF(t1<=0);
               !!! Error reading: skip;
               #SKIP;
          #END;

          !!! Get BUF for area [c1] ;
          buf=TBL_ITM(iepT,£BUF,c1);
          !!! Read I16 at start of area ;
          t1=BUF_VAL(buf,1,NULL,£i16);
          !!! Changed ? ;
          #IF(t1!=TBL_ITM(iepT,£oldCnt,c1));
               TBL_ITM(iepT,£oldCnt,c1,t1);
               !!! Write clock in area at address 100 (0bsd) ;
               BUF_VAL(buf,101,CLOCK,£U32);
               !!! Write only U32;
               IEP_CLIWR(iep,101,4);
          #END;
     #END;
#END;

!!! Trash the 4 iep and 4 buf here ;
c1=0;
#WHILE(c1+=1 <=4);
     TRASH(TBL_ITM(iepT,£IEP,c1));
     TRASH(TBL_ITM(iepT,£BUF,c1));
#END;
```

EXAMPLE.PKG.1	
Package background image substitution	
In the program package, in the Resources section, there is a BKG element that refers to a background "bkg1.png". We want to replace it with a new "bkg2.png" background.	
ATTRIBUTES	
`pkgPtr`	Pointer to package
`pkgPth`	Path of the package
`t1`	Temporary attrib
CODE	

```
!!! Open package ;
pkgPtr=PKG_OPN("c:\PWK-PRG\stressGui.pkw");

!!! Get package pointer ;
pkgPth=PKG_PTH(pkgPtr);

!!! Get old file name ;
t1=PTH_MNFGET(pkgPth,£Resources,£BKG);

!!! Change name of the background file in manifest ;
PTH_MNFADD(pkgPth,£Resources,£BKG,"bkg2.png");

!!! Copy file from actual position into the package ;
FS_FCPY("c:\draw\bkg2.png",pkgPth++"bkg2.png");

!!! Del old file from package ;
FS_FDEL(pkgPth++t1);

!!! Save package ;
PKG_SAV(pkgPtr,£Resources,£BKG);
```

EXAMPLE.MTH.1	
Calculate interpolation of coordinates	
In the program package, in the Resources section, there is a BKG element that refers to a background "bkg1.png". We want to replace it with a new "bkg2.png" background.Given a series of coordinates, interpolation is carried out to calculate the value of Y at points other than those provided as a reference. The starting point is a table with X and Y columns with the coordinates to be interpolated.	
ATTRIBUTES	
`coordT`	TBL with coordinates
`PLY`	Pointer to polynomial
`xInp`	x coordinate input
`yOut`	y coordinate result
CODE	

```
!!! TBL creation from column names and row number ;
coordT= TBL_NEW(NULL,5,";",NULL,"X;Y");

!!! TBL population : x1,y1,x2,y2,x3,y3,x4,y4,x5,y5 ;
TBL_ITM(coordT,£X,1,"ROW 2",10,5.2,20,7.4,30,11.1,40,8.5,50,6.2);

!!! Create a polynomial of 5th degree ;
PLY= MTH_PLY(NULL,coordT,5);   !!! Default xCol=£X, yCol=£Y,elem.n by the TBL ;

yOut= MTH_PLY(PLY,£Y,xInp);
```